

CompactFlash (CF) Interface

For MCS-BASIC52 systems

Please note:

Readers' Circuits have not been tested or post-engineered by the Elektor Electronics laboratory.

Design by G. Meers

Prices of CompactFlash cards having dropped considerably over the past year or so, the author decided to design a parallel PC interface that allows data or programs to be written into one of these handy solid-state memories using MCS-BASIC52.

CompactFlash (CF) cards are gaining popularity every day, probably because the vast majority of digital cameras employ them for mass

data storage, and, in a more general sense, because the phenomenon 'digital camera' itself is taking the world

by storm. Thanks to ongoing developments in CF technology, the memory capacity rises while prices drop. This

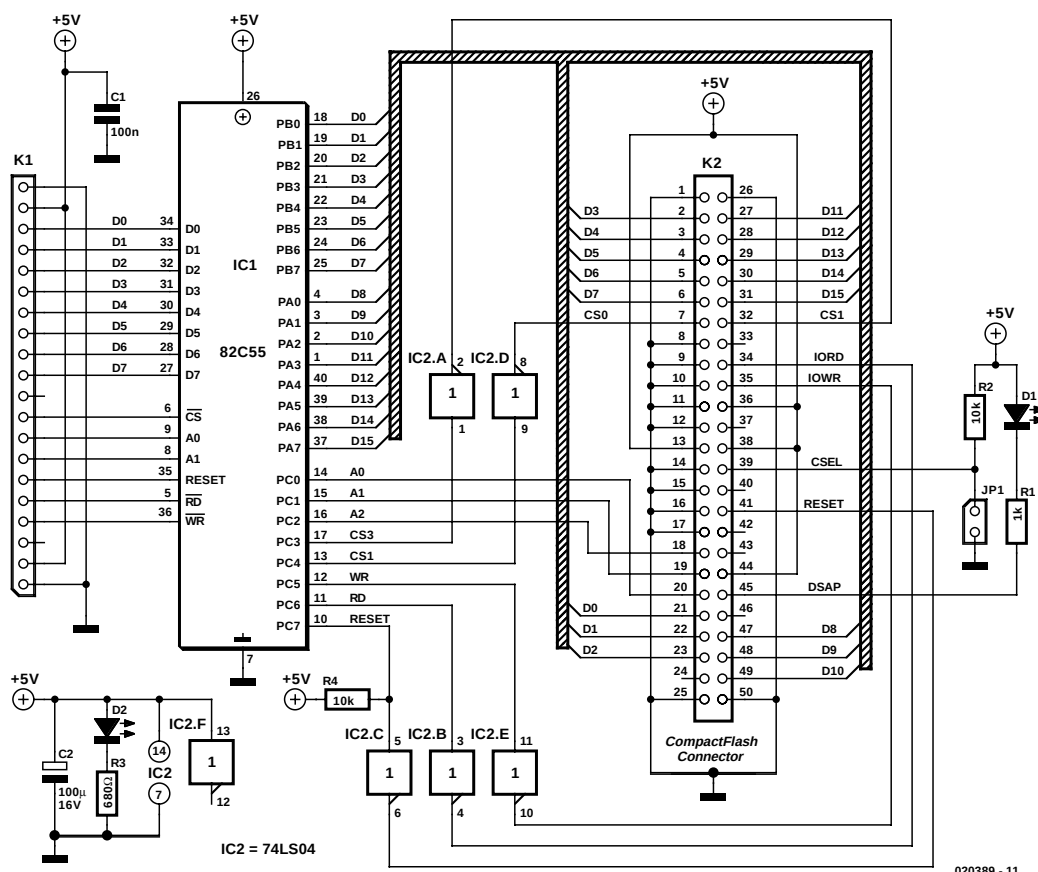


Figure 1. The required hardware is down to a minimum.

makes the CompactFlash card particularly attractive for PC users too, and you should not have been surprised by us publishing a CompactFlash-to-IDE Interface as early as April 2002.

Triggered by the above mentioned publication, the author set out to design a CF interface for connection to the address bus, data bus and control bus of the PC. The resultant circuit, in combination with some simple software, allows BASIC52 programs to be stored on a CompactFlash card.

The software developed for this project may be downloaded free of charge from our website, www.elektor-electronics.co.uk as file number **020389-11.zip** which may be found under month of publication. The program reports attempts to enter incorrect parameters or undesirable write actions to memory blocks. The interface draws a current of about 42 mA and achieves a data throughput rate of about 50 kB per second. The author actually employs the interface on a 20-MHz system with excellent results and reliable data transfers.

Circuit diagram

The circuit diagram of the CF Interface is shown in **Figure 1**. As you can see, the required hardware remains limited to an 82C55 chip, a 74LS04, two indicator LEDs and two connectors. With some luck, many of the components may be found in the junkbox.

A 50-way connector, K2, allows the CF card to be inserted. On this connector, pin 9 is wired to ground, forcing the card to behave like an IDE hard disk. In this mode of operation, jumper JP1 (between pin 39 and ground) **must** be fitted.

The complete control of the IDE interface is handled by the 82C55 (IC1). Using its bi-directional ports PB0-PB7, the chip drives data lines D0-D7, while port lines PA0-PA7 handle data lines D8-D15. Port lines PC0-PC7 are programmed to look after the control lines that form part of the IDE interface. Using software, bi-directional ports PA0-PA7 and PB0-PB7 are set up as inputs or outputs at the proper instant. By contrast, port lines PC0-PC7 are always configured as outputs.

The inverters contained in IC2 are

needed because the 82C55 pulls the relevant port lines Low when the data direction is reversed. When this takes place, the inverters prevent contention and undefined levels on the control lines.

The following signals have to be connected to the system by way of connector K1: 5 V and ground; D0-D8; RD; WR; A0, A1 and RESET; CS.

Note that the address decoder for CS (Chip Select) is **not** included in the circuit. The CS signal has to go Low when addresses from 0C000h are selected by software. The 82C55 requires four addresses:

DATA register PA, base address +0;
DATA register PB, base address +1;
DATA register PC, base address +2;
CONTROL register, base address +3.

A few more circuit details: LED D1 lights to indicate activity on the CompactFlash card. D2 indicates the presence of the supply voltage. Finally, capacitors C1 and C2 are included to decouple (filter) the supply voltage.

Software

The software for the interface was written in ASM51 and the source code may be found under two names:

- PARCFFU.ASM (for PCFF commands from MCS-BASIC52);
- PARCFF.ASM (for CALL commands from MCA-BASIC52).

The software comprises a rudimentary IDE driver that divides the CompactFlash card into blocks of 8,192 bytes. A cheap 64 MB CF card has a storage capacity of 7,828 blocks, or 64,086,016 bytes. That's slightly less than the nominal capacity of the card because the last virtual cylinder is not completely used in the software.

In the program source code, the 82C55 has its addresses assigned as follows:

```
P8255A EQU 0C008H ; PORT A ADDRESS
P8255B EQU 0C009H ; PORT B ADDRESS
P8255C EQU 0C00AH ; PORT C ADDRESS
P8255CO EQU 0C00BH ; CONTROL ADDRESS
```

If the CS signal on connector K1 selects other addresses, then the above allocations should be changed accordingly.

Because the software is intended as an extension of MCS-BASIC52 the system must provide for sufficient extra program memory starting at address 2000h. The program proper has a size of 1.6 kB. In case you've already extended the system with additional MCS-BASIC52 commands, obviously the source code has to be adapted and re-assembled. When doing so, bear in mind that the source code employs ACALL instructions with an addressing limit of 2 kB!

Finally, the software for the CF Interface is compatible with MCS-BASIC52 V1.1 and V1.3.

PARCFFU.ASM

This is the source code of the program you'll be using to extend MCS-BASIC52 with extra statements. The assembled (object) code called PARCFFU.HEX has to be copied into the system memory starting at location 2000h.

The following commands are then available:

- **PCFF V** – Show version of connected CompactFlash card.
- **PCFF R** – Reset CompactFlash card.
- **PCFF L, block number, address** – Load data from the CompactFlash card into the system. The 'block number' is the location on the card (0-7823) and has a fixed size of 8 kB. The parameter 'address' contains the start address of the system data memory (00000h through 0FFFFh). After each processed sector, an 'L' is sent to the terminal.
- **PCFF S, block number, address** – Save data from the system data memory into the CompactFlash card. As above, only an 'S' is sent after each sector.
- **PCFF F, block number** – This causes the complete contents of this block number on the card to be filled with the value 0FFh. An 'F' is sent on completion of a sector.
- **PCFF D, block number** – Show status of block number: B = data; V = empty.

Unambiguous error reports will pop up if the card is either not ready, the block number is too high or if the block is in danger of being overwritten.

Using the PCFF instructions

Programs developed under MCS-BASIC are contained in the RAM memory from address 0200h onwards. The instruction PCFF S, 10, 0200H writes the program to block 10 on the

card. Using PCFF L, 10, 0200H the program may be placed into RAM memory again.

If the program is larger than 8 kB (find out using P. LEN) it has to be accommodated in multiple blocks. If this is required, you should note the following. The instruction PCFF S, 10, 0200H writes data from address range 0200H through 21FDH into block 10. This represents a total of 8,190 bytes, not 8,192 as you may have expected because two bytes are used to flag that the block already contains data. For example, to write (save) a program of 10 kB, the following instructions should be used:

```
PCFF S, 10, 0200H : PCFF S, 11, 21FEH
```

or

```
PCFF S, 10, 0200H : PCFF S, 11, 0200H+8190
```

Reloading the program is then done as follows:

```
PCFF L, 10, 0200H : PCFF L, 11, 0200H+8190
```

Variables may be used for the block number and the address. If you want to erase, say,

block 100 to 120, the following little program may be run:

```
100    FOR T=100 to 120
101    PCFF F, T
102    NEXT T
103    END
```

PARCFF.ASM

In some cases, it is not possible to extend the system with extra MCS-BASIC52 statements starting at program address 2000H. That's why the source code contained in PARCFF.ASM is intended to enable the program to be fitted at any location in the system's program memory. In this source code, the start address is 0B000h and this may be changed to your requirements. The start address is defined using the following code chunk:

```
START EQU 0B000H
ORG  START
```

This allocation may be changed and a new program address created. Obviously, the source code then has to be assembled again.

The operation and practical use is identical with PARCFFU.ASM described above, only the letters 'PCFF' in the instructions have to be replaced by 'CALL 0B000H'. Information on the way the necessary variables are entered in assembly code may be found in the source code file.

(020389-1)

Free Download

– ASM51 source code and HEX files of PARCFFU and PARCFF.
File number: **020389-11.zip**.

www.elektor-electronics.co.uk/dldl.htm

(select month of publication).